

## **Rapport, Algorithmique 2**

### Liste de mots de lexiques donnés

#### **L'objet de chacun des modules**

1. `hashtable`, `holdall` → modules déjà donnés en CM et aussi en TP, nous les utilisons sans les modifier pour stocker les mots et leur fichiers associés ainsi que leur numéros de lignes.
2. `listdyn` → module utilisé pour l'implantation des fonctions sur les listes dynamiques simplement chaînées avec pointeur de tête et de queue et des valeurs de type `void *`.
3. `list_line` → module utilisé pour l'implantation des fonctions sur les listes dynamiques simplement chaînées avec pointeur de tête et de queue et des valeurs de type `long int`.
4. `word_implantation` → module utilisé pour l'implantation des fonctions de lecture de mot dans les fichiers et sur l'entrée standard, ainsi que des fonction d'insertions dans les différentes tables de hachage et pour finir une fonction d'affichage sur la sortie standard ou dans un fichier et une fonction de teste sur des caractères.
5. `main` → fichier contenant le programme principale exécutable `lidx.c` avec implantation des mots récupérés sur la ligne de commande, utilisations des fonctions contenue dans le module `word_implantation` et libération de la mémoire.

#### **Déscription de l'implantation**

Dans le fichier `main.c` se trouve l'exécutable `lidx.c` dans lequel nous utilisons les fonctions définies dans le fichier header `word_implantation`.

Nous utilisons des listes dynamiques simplement chaîné de type `listdyn` pour stocker tout les noms de fichiers associés a chaque mot et d'autres listes de type `list_line` pour associer les mots trouvés par le lexique finale aux lignes où ils ont été trouvé. Nous utilisons aussi deux tables de hachage initialement vide avec l'aide de la fonction `hashtable_empty`, qui prend en paramètres la fonction de comparaison pour les chaînes de caractères (`strcmp`) et la fonction de pré-hachage (`str_hashfun`). La table `ht` est utilisée pour associer les mots (les clés) avec leurs provenance (noms de fichiers) et `ht_num` pour associer des mots (les clés) avec leurs numéros de ligne de lecture. Nous utilisons deux fourretouts, `haword` qui stocke tous les mots différents et `hafname` qui stocke les noms de fichiers lu en arguments.

Quand un fichier est trouvé comme lexique sur la ligne de commande nous utilisons la fonction `word_read_argv` qui prend en arguments la table de hachage `ht`, le fourretout `hafname`, le fourretout `haword`, le nom du fichier et l'option `up_low`. Cette fonction lit chaque mot des lexiques donnés et utilise une autre fonction `insertion_hashtable` qui prend en arguments la table de hachage `ht`, les holdall `haword` et `hafname` et le nom du fichier et qui insère les mots dans `ht` et dans `haword` si celui-ci n'est pas dedans.

Si un mot ou une phrase est lu sur la ligne de commande alors la lecture du mot se fait dans le fichier `lidx.c` et l'insertion dans la table de hachage `ht` et dans le holdall `haword` est réalisée également par la fonction `insertion_hashtable` qui prends les mêmes arguments que précédemment mais avec comme nom de fichier : "string".

Une fois que chaque argument est lu, un appel de la fonction `word_read_stdin` est réalisé. Cette fonction prend comme argument la table de hachage `ht_num`, le holdall `haword` et l'option `up_low`. Elle lit les mots du lexique finale (celui après le symbole '<'), insert les numéros de ligne dans `ht_num` si le mot est déjà dans `haword` et donc si celui-ci était dans au moins un lexique. Cette insertion se fait grâce à la fonction `insertion_hashtable_line` qui prend en arguments la table de hachage `ht`, le mot lu et le numéro de ligne de lecture du mot.

La fonction d'affichage `print` prends en arguments les deux tables de hachage (`ht`, `ht_num`), les deux holdall (`hafname`, `haword`), un tableau de pointeurs sur les mots entrés sur la ligne de commande, les options d'affichage et de tri (ou non) des mots. La fonction affiche d'abord les mots lu sur la ligne de commande puis cherche chaque fichier dans la table de hachage `ht` et affiche 'X' si celui-ci est dans la liste associé au mot et rien sinon, puis affiche chaque numéro de ligne à laquelle le mot apparaît dans le lexique final avec "¶" à la fin, sinon "¶" après chaque affichage de ligne.

Les fonctions `listdyn_remove`, `listdyn_dispose` et `list_line_dispose` désallouent et suppriment succesivement les mots du holdall `haword`, les listes associé au mot dans la table de hachage `ht` et dans `ht_num`. Une fois l'affichage terminé un goto dispose permet de désallouer les tables de hachages et les fourretouts.

## Exemples de lignes de commande et d'affichages

→ `./lidx Un - lex0.txt -o ouput.txt < lex2.txt`



→ ./lidx - lex0.txt Un -S -u -o output.txt < lex2.txt

```

Terminal - alexis@alexis: ~/algo2.2/projet/main
Fichier  Édition  Affichage  Terminal  Onglets  Aide
alexis@alexis:~/algo2.2/projet/main$ ./lidx - lex0.txt "Un" -S -u < lex2.txt
lex0.txt
COMPTE          X
DOIGTS          X
MES             X
SUR             X
UN              X
alexis@alexis:~/algo2.2/projet/main$

```

→ ./lidx salut < tartuffe.txt

```

Terminal - alexis@alexis: ~/algo2.2/projet/main
Fichier  Édition  Affichage  Terminal  Onglets  Aide
alexis@alexis:~/algo2.2/projet/main$ ./lidx salut < tartuffe.txt
salut      X      190, 399, 1701, 1755 ¶
alexis@alexis:~/algo2.2/projet/main$

```

→ time ./lidx salut - lex0.txt - lex1.txt -S -u < tartuffe.txt

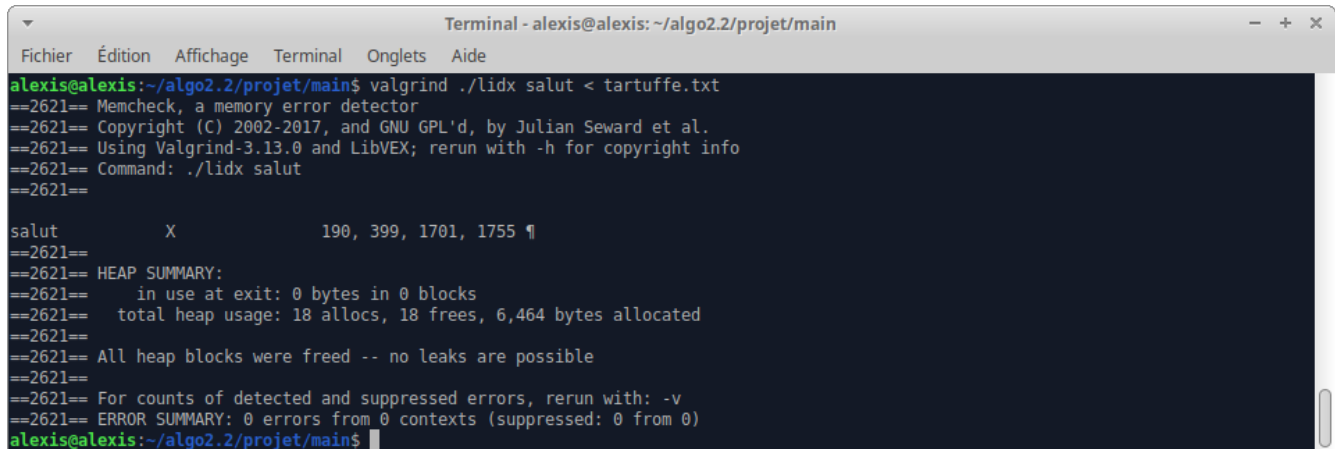
```

Terminal - alexis@alexis: ~/algo2.2/projet/main
Fichier  Édition  Affichage  Terminal  Onglets  Aide
alexis@alexis:~/algo2.2/projet/main$ time ./lidx salut - lex0.txt - lex1.txt -S -u < tartuffe.txt
lex0.txt      lex1.txt
AMIS          X      43, 342, 1522 ¶
AMOURS        X      ¶
COMPTE        X      ¶
DOIGTS        X      1703 ¶
EMMERDES      X      ¶
MES           X      X      43, 63, 225, 616, 726, 960, 1126, 1172, 1225, 1237, 1319, 1333, 1359,
1402, 1555, 1666, 1731, 1762, 1787, 1788, 1863, 1901, 2232, 2255, 2401, 2448, 2476, 2491, 2497, 2751, 2752, 2760, 2762, 2763, 2796, 28
02, 2822, 2853, 2860, 2976, 2983 ¶
SALUT         X      190, 399, 1701, 1755, 2901 ¶
SUR           X      57, 76, 108, 195, 233, 270, 294, 314, 353, 443, 465, 767, 864, 988, 11
11, 1141, 1319, 1466, 1470, 1555, 1570, 1575, 1625, 1711, 1746, 1772, 1805, 1820, 1902, 1935, 2049, 2242, 2277, 2432, 2472, 2482, 2515
, 2690, 2807, 2819, 2847, 2951, 2991, 2993, 3074, 3161 ¶

real    0m0,013s
user    0m0,012s
sys     0m0,000s
alexis@alexis:~/algo2.2/projet/main$

```

→ valgrind “salut ca va” < tartuffe.txt



```
Terminal - alexis@alexis: ~/algo2.2/projet/main
Fichier  Édition  Affichage  Terminal  Onglets  Aide
alexis@alexis:~/algo2.2/projet/main$ valgrind ./lidx salut < tartuffe.txt
==2621== Memcheck, a memory error detector
==2621== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2621== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==2621== Command: ./lidx salut
==2621==
salut      X      190, 399, 1701, 1755 ¶
==2621==
==2621== HEAP SUMMARY:
==2621==   in use at exit: 0 bytes in 0 blocks
==2621==   total heap usage: 18 allocs, 18 frees, 6,464 bytes allocated
==2621==
==2621== All heap blocks were freed -- no leaks are possible
==2621==
==2621== For counts of detected and suppressed errors, rerun with: -v
==2621== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
alexis@alexis:~/algo2.2/projet/main$
```

## Problèmes

Nous n’avons pas implémenté l’option *input* et nous avons aussi des problèmes de désallocations, le holdall *haword* est mal libéré quand l’option -S n’est pas demandé. En effet nous n’avons pas réussi à désallouer (*dans ce cas*) les *char \** de chaque éléments du *haword* sans casser l’affichage. (la solution était de modifier le *lisdyn\_remove* et d’y ajouter *free(t → value)*).

*Remarque : J’ai délibérément laisser les ponctuation entre 2 mots non séparés pas un espace pour ne pas couper les mots tels que peut-etre, aujourd’hui, 1/2 etc....*